

Into the Square

On the Complexity of Quadratic-Time Solvable Problems

Michele Borassi¹, Pierluigi Crescenzi², and Michel Habib^{3,4}

¹IMT Institute for Advanced Studies, 55100 Lucca, Italy,
michele.borassi@imtlucca.it

²Università di Firenze, Dipartimento di Ingegneria dell'Informazione, 50134 Firenze,
 Italy, pierluigi.crescenzi@unifi.it

³LIAFA, UMR 7089 CNRS & Université Paris Diderot, France,
habib@liafa.univ-paris-diderot.fr

July 21, 2014

Abstract

In this paper, we will analyze several quadratic-time solvable problems, and we will classify them into two classes: problems that are solvable in *truly subquadratic* time (that is, in time $\mathcal{O}(n^{2-\epsilon})$ for some $\epsilon > 0$) and problems that are not, unless the well known Strong Exponential Time Hypothesis (in short, SETH) is false. In particular, we will prove that some quadratic-time solvable problems are indeed easier than expected. We will provide an algorithm that computes the transitive closure of a directed graph in time $\mathcal{O}(mn^{\frac{\omega+1}{4}})$, where m denotes the number of edges in the transitive closure and ω is the exponent for matrix multiplication. As a side effect of our analysis, we will be able to prove that our algorithm runs in time $\mathcal{O}(n^{\frac{5}{3}})$ if the transitive closure of the graph is sparse. The same time bounds hold if we want to check whether a graph is transitive, by replacing m with the number of edges in the graph itself. As far as we know, this gives us the fastest algorithm for checking whether a sparse graph is transitive. Finally, we will also apply our algorithm to the comparability graph recognition problem (which dates back to 1941): also in this case, we will obtain the first truly subquadratic algorithm. In the second part of the paper we will deal with hardness results. In particular, we will start from an artificial quadratic-time solvable variation of the k -SAT problem and we will construct a graph of Karp reductions, proving that a truly subquadratic-time algorithm for any of the problems in the graph falsifies SETH. More specifically, the analyzed problems are the following: computing the subset graph, finding dominating sets, computing the betweenness centrality of a vertex, computing the minimum closeness centrality, and computing the hyperbolicity of a pair of vertices. We will also be able to include in our framework three proofs already appeared in the literature, concerning the problems of distinguishing between split graphs of diameter 2 and diameter 3, of solving the local alignment of strings and of finding two orthogonal binary vectors inside a collection.

1 Introduction

Since the very beginning of theoretical computer science and until recent years, the duality between NP-hard problems and polynomial-time solvable problems has been considered the threshold distinguishing “easy” from “hard” problems. However, polynomial-time algorithms might not be as efficient as one expects: for instance, in real-world networks with millions or billions of nodes, also quadratic-time algorithms might turn out to be too slow in practice, and a *truly subquadratic* algorithm would be a significant improvement, where an algorithm is said to be truly subquadratic if its time-complexity is $\mathcal{O}(n^{2-\epsilon})$ for some $\epsilon > 0$.

1.1 Subquadratic-Time Results

In the first part of this paper, we will analyze two well-known problems (that is, checking whether a graph is transitive and recognizing comparability graphs) and we will show that they are solvable in truly subquadratic time. For what concerns the transitivity problem, our main contribution is a new analysis of an old algorithm that finds the transitive closure of a graph [26]. This analysis will lead us to a simple modification of the algorithm itself, that will provide an $\mathcal{O}\left(mn^{\frac{\omega+1}{4}}\right)$ algorithm, where m is the number of edges in the transitive closure and ω is the exponent for matrix multiplication, whose current value is 2.3727 [44]. As a consequence, we will be able to check if a graph is transitive in truly subquadratic time: as far as we know, no truly subquadratic algorithm was previously known for this problem, although many papers have been published on the computation of the transitive closure [11, 7, 34]. As a side effect of our analysis, we will be able to prove that our algorithm runs in time $\mathcal{O}(n^{\frac{5}{3}})$ in graphs whose transitive closure is sparse: as far as we know, this gives us the fastest algorithm for checking whether a sparse graph is transitive. More importantly, combining this result with the results in [28], we will be able to provide a truly subquadratic algorithm for recognizing comparability graphs, a widely studied graph class (for more information on comparability graphs, we refer to [10]). As far as we know, the existence of such an algorithm was also not known before our result, even if this class of graphs is quite old (the oldest mention we were able to find dates back to 1941 [17]).

1.2 Hardness Results

Following the main ideas behind the theory of NP-completeness, and not being able to show that a specific polynomial-time solvable problem might or might not admit a faster algorithm, researchers have recently started to prove that the existence of such an algorithm would imply faster solution for many other problems. As an example, a huge amount of work started from the analysis of the 3SUM problem, which consists of, given three sets A , B and C of integers, deciding whether there exists $a \in A$, $b \in B$, and $c \in C$ such that $a + b + c = 0$. This problem has been widely studied and, as far as we know, the best algorithm has been provided in [4]: this algorithm is subquadratic, but not truly subquadratic. The 3SUM problem has then become a starting point for proving the “hardness” of many other problems, especially in algebraic geometry (for example, we refer the interested reader to [31]). Notice that all these results do not deal with the notion of completeness, but they simply prove that “a problem is harder than another”, relying on the fact that the easiest problem has been studied for years and no efficient algorithm has been found.

A more recent develop of this field is based on the Strong Exponential Time Hypothesis (SETH), used as a tool to prove the hardness of polynomial-time solvable problems. This hypothesis, stated in [29], says that there is no algorithm for solving the k -SAT problem in time $\mathcal{O}((2 - \epsilon)^n)$, where $\epsilon > 0$ does not depend on k . Successively, researchers have started to use it in order to prove hardness results (see for example [45, 37], where the authors address the hardness of many problems, like the all-pairs-shortest-paths, finding triangles in a graph, 2-SAT,

k -dominating set, and some generalization of matrix multiplication). Starting from these works, many other hardness results based on SETH have been published, and many of them deal with dynamic problems (see, for instance, [1]). As an example, it is worth mentioning the diameter computation, that is, given a graph, finding the maximum distance between two vertices. Despite numerous papers on the topic, no truly subquadratic algorithm has been found so far (for more details on the diameter computation, we refer to [42, 41, 16, 8]). However, a reason for this behavior was found in [39], where it is proved that a truly subquadratic algorithm to distinguish graphs of diameter 2 and 3 would falsify SETH (see also [40]). This result is meaningful both because it gives a lower bound on the complexity of the diameter problem, and because it sheds some light on SETH, which is becoming more and more central in modern computer science. Other similar result are the hardness of the local sequence alignment problem, proved in [2], and the hardness of finding two orthogonal binary vectors in a collection [43].

On the ground of this approach, in the second part of this paper we will show that several well-known quadratic-time solvable problems are not solvable in truly subquadratic time, unless SETH is false. As a first step, we will define the problem k -SAT* (obtained through an “artificial” modification of the input of the k -SAT problem), which cannot be solved in truly subquadratic time, unless SETH is false (note that the use of the k -SAT* problem has already been implicitly suggested in [45]). We will then design several Karp-reductions that preserve truly subquadratic-time resolvability, and we will use these reductions in order to prove the hardness of several other problems (note that all problems considered, apart from the 3-dominating set, are actually solvable in quadratic time, so that the lower and upper bounds coincide, apart from logarithmic factors). More specifically, the analyzed problems are the following.

SUBSETGRAPH: given a collection \mathcal{C} of subsets of a given ground set X , find the subset graph of \mathcal{C} . The subset graph is defined as a graph whose vertices are the elements of \mathcal{C} , and containing an edge (C, C') if $C \subseteq C'$. For this problem, the first subquadratic algorithm was proposed in [47], and in [38, 20] matching lower bounds are proved. However, these lower bounds are based on the number of edges in the subset graph, which might be quadratic with respect to the input size, apart from logarithmic factors. Our results show that the complexity of computing the subset graph is not due to the output size only, but it is intrinsic: in particular, we will prove that even deciding whether the subset graph has no edge is hard. This excludes the existence of a truly subquadratic algorithm to check if a solution is correct, or a truly subquadratic algorithm for instances where the output is sparse.

BETWEENNESSCENTRALITYVERTEX: the betweenness centrality of a vertex in a graph is a widely used graph parameter related to community structures, defined in [25] (for more information we refer to the book [36] and the references therein). Despite numerous attempts like [9, 3, 18], there exist no truly subquadratic algorithm computing the betweenness centrality, even of a single vertex. Moreover, in [3], it is said that finding better results for approximating the betweenness centrality of all vertices (BETWEENNESSCENTRALITY) is a “challenging open problem”. Our analysis does not only prove that computing the betweenness centrality of all vertices in subquadratic time is against SETH, but it also presents the same result for computing the betweenness of a single vertex.

MINIMUMCLOSENESSCENTRALITY: another fundamental parameter in graph analysis is closeness centrality, defined for the first time in 1950 [5] and recently reconsidered when analyzing real-world networks (for the interested reader, we refer to [33] and the references therein). This problem has also raised algorithmic interest, and the most recent result is a very fast algorithm to approximate the closeness centrality of all vertices [14]. In this paper, we will prove for the first time the hardness of finding the “least central” vertex with respect to this measure. Simple consequences of this results are the hardness of computing

in truly subquadratic time the closeness centrality of all vertices, or of extracting a “small enough” set containing all peripheral vertices.

HYPERBOLICITYWITH2FIXEDVERTICES: the Gromov hyperbolicity of a graph [27] is another parameter that recently got the attention of researchers in the field of network analysis. This parameter has relations with the chordality of a graph [46], and with diameter and radius computation [12, 16]. In [15], it is provided an algorithm with time-complexity $\mathcal{O}(n^4)$, but much more efficient in practice, while in [23] an algorithm with time-complexity $\mathcal{O}(n^{3.69})$ is described. The latter paper also provides an algorithm with complexity $\mathcal{O}(n^{2.69})$ to compute the hyperbolicity when a vertex is fixed. One might think that if two vertices are fixed, then the bound becomes subquadratic: we will prove that this is not true unless SETH is false. This is a significant result on its own right, but it can also be used in the design of algorithms that compute the hyperbolicity of the whole graph.

3DOMINATINGSET: the last “classical” problem analyzed is finding dominating sets in a graph, which is one of the 21 Karp’s NP-complete problems [30]. Researchers have mainly tried to efficiently determine a k -dominating set when k is fixed [19]. Also negative results were published on the minimum dominating set problem: in [37] it is proved that finding a dominating set of size k in $\mathcal{O}(n^{k-\epsilon})$ would falsify SETH. Our paper proves that a subquadratic-time algorithm for the 3-dominating set problem would falsify SETH. This result is complementary to the result of [37]: their result works better on dense graphs, while our result works better on sparse graphs. We will also prove that, given a bipartite graph $G = (V, W, E)$, finding a pair of vertices in V that dominates W is hard.

We will also be able to include in our framework three proofs already appeared in the literature, concerning the problems of distinguishing between split graphs of diameter 2 and diameter 3 [39], of solving the local alignment of sequences [2], and of finding orthogonal vectors in a collection [22].

In order to prove our reductions, we have also analyzed several other problems, that play the role of “intermediate steps”. These intermediate steps are very natural problems, and they can be considered as significant results on their own right. For instance, many of these problems deal with a collection \mathcal{C} of subsets of a given ground set X , and try to find particular pairs in the collection (TWO DISJOINT SETS, SPERNER FAMILY, TWO COVERING). These problems are very natural, but we have not been able to find any paper on the topic, apart from the SPERNER FAMILY case, that is, finding whether there exists two sets $C \subseteq C'$ in the collection \mathcal{C} . Such a family, also named clutter, has been analyzed in many cases (see for example [21, 6]), but no recognition algorithm has been provided, yet. For each of these set problems, this paper deals with two variations: in the first one, the size of the collection is exponential with respect to the size of X , while in the other one any size is accepted (so the problem becomes harder). Other problems have been obtained by “rephrasing” these problems in terms of graphs, for example finding dominated vertices (BIPGRAPH DOMINATED VERTEX and GRAPH DOMINATED VERTEX). We believe that these results are interesting in their own right, but they also can be used as intermediate points for new reductions, without restarting from SETH for any new hardness proof.

1.3 Structure of the Paper

In Section 2 we will discuss the two problems that, although difficult at first glance, actually can be solved in truly subquadratic time. In Section 3 we will provide the framework in which all hardness results are given and we will state and prove all reductions. Section 4 concludes the paper and provides some open problems.

2 Transitive Closure and Comparability Graph Test

In this section, we will prove that the complexity of recognizing transitive directed graphs is truly subquadratic. As a result, we will also be able to provide a truly subquadratic algorithm to recognize comparability graphs.

Let us start by assuming without loss of generality that the input graph is acyclic, since the transitive closure of a connected component is a clique, and it is possible to find connected components in linear time.

The new algorithm on acyclic graphs relies heavily on a new analysis of the old algorithm published in [26] (the pseudo-code is given in Algorithm 1). This analysis is provided by the following theorem.

Theorem 2.1. *Let α be any real value in $[0, 1]$. Algorithm 1 finds the transitive closure of the input graph G in time $\mathcal{O}(\frac{m^2}{n^{2\alpha-1}} + mn^\alpha)$, where m is the number of edges in the transitive closure.*

Before proving the theorem, we will apply it in the design of truly subquadratic algorithms, both for checking transitivity closure and for comparability graph recognition. Note that the algorithm for sparse graphs is practical, since it is purely combinatoric and it does not rely on matrix multiplication like the one in [7].

Corollary 2.2. *If the transitive closure of a graph is sparse, then it can be computed in time $\mathcal{O}(n^{\frac{5}{3}})$.*

Proof. Apply the previous theorem with $\alpha = \frac{2}{3}$. □

In order to obtain a truly subquadratic algorithm for any graph, we need to pair the previous algorithm with matrix multiplication: as a consequence, the algorithm in the following corollary might have huge constants hidden inside the \mathcal{O} notation, differently from the previous one.

Corollary 2.3. *It is possible to compute the transitive closure of any graph in time $\mathcal{O}(mn^{\frac{\omega+1}{4}})$, where m is the number of edges in the transitive closure.*

Proof. In [32], Algorithm 5.2, it is provided an algorithm that computes the transitive closure in time $\mathcal{O}(n^\omega)$. We will use this algorithm if $m \geq n^{\frac{3\omega-1}{4}}$, and the time complexity is $\mathcal{O}(n^\omega) \leq (mn^{\frac{\omega+1}{4}})$. Otherwise, that is, if $m < n^{\frac{3\omega-1}{4}}$, we will use Algorithm 1: applying Theorem 2.1 with $\alpha = \frac{\omega+1}{4}$, we obtain a time complexity of $\mathcal{O}\left(\frac{m^2}{n^{2\alpha-1}} + mn^\alpha\right) = \mathcal{O}\left(m^{\frac{3\omega-1}{2}} + mn^{\frac{\omega+1}{4}}\right) = \mathcal{O}\left(mn^{\frac{\omega+1}{4}}\right)$.

However, the value of m is not known in advance: in any case, it is enough to start by applying Algorithm 1 until the number of edges reaches n^α , and continue with matrix multiplication if this is the case. □

The following corollary applies the previous results to transitive graph recognition.

Corollary 2.4. *It is possible to check if a graph is transitive in time $\mathcal{O}(\frac{m^2}{n^{2\alpha-1}} + mn^\alpha)$, and $\mathcal{O}(n^{\frac{5}{3}})$ if the graph is sparse.*

Proof. It is easy to see that if the previous algorithms are stopped as soon as a single edge is added, the running time bounds depend on the input size and not on the output size. □

Finally, next corollary will apply the previous algorithms to the comparability graph recognition problem, providing again the first truly subquadratic algorithm for this task.

Corollary 2.5. *It is possible to check if a graph is a comparability graph in time $\mathcal{O}\left(mn^{\frac{\omega+1}{4}}\right)$. If the graph is sparse, the running time is $\mathcal{O}\left(n^{\frac{5}{3}}\right)$*

Proof. It is known that, if a graph G is a comparability graph, then it is possible to compute a transitive orientation of G in linear time [35] (for a simpler algorithm running in $\mathcal{O}(n + m \log n)$ see [28]). With input a graph G , our comparability test runs the transitive orientation algorithm: let H be the output of this algorithm. Then, we use the previous transitive closure algorithm (either the one for sparse graphs or the general one), in order to decide if H is transitive. If it is, G is clearly a comparability graph. If H is not transitive, it means that the graph G is not a comparability graph, since otherwise the transitive orientation algorithm would have provided a transitive orientation.

The running time of comparability graph recognition coincides with the running time of the transitive closure algorithm, since all other steps are faster. This proves the theorem. \square

The rest of this section is devoted to the proof of Theorem 2.1, which analyzes the complexity of Algorithm 1.

```

1 TransClosure( $G = (V, N)$ ) {
2   find a topological ordering  $(\sigma_1, \dots, \sigma_n)$  of  $V$ ;
3   for (int  $i = n$ ;  $i > 0$ ;  $i--$ ) {
4      $N'(\sigma_i) = N(\sigma_i)$ ;
5     for  $w$  in  $N_i(v)$  {
6        $N'(\sigma_i) = N'(\sigma_i) \cup N'(w)$ 
7     }
8   }
9   return  $(V, N')$ 
10 }
```

Algorithm 1: Computing the transitive closure of a graph.

We will divide the proof into a sequence of lemmas. For completeness, we also include the correctness proof (Lemmas 2.6 and 2.7), already provided in [26]. First of all, let us restate the definition of transitive closed graph, so that it is easily checkable.

Lemma 2.6. *A directed acyclic graph is transitive if and only if for each edge (v, w) , $N(v) \supseteq N(w)$.*

Proof. If the graph is transitive, and $x \in N(w)$, then there is the path (v, w, x) , and by transitivity $x \in N(v)$. For the other direction, let us assume there is a path from v to w and let us prove that v is linked to w by induction on the length k of the path (the base case holds by hypothesis). For induction step, let v_1 be the first vertex of the path after v : v_1 is linked to w by inductive hypothesis, and we conclude because $N(v) \supseteq N(v_1)$. \square

Lemma 2.7. *The algorithm is correct, that is, the output graph is the transitive closure of the input graph.*

Proof. We will prove by backward induction that after vertex σ_i is analyzed in the **for** cycle at line 4, the condition in Lemma 2.6 is verified for each pair (σ_j, σ_k) with $j \geq i$, for neighbors N' .

The base step is trivial, since σ_n has no outgoing edge. For induction step, if $j > i$ there is nothing to prove, since the only modified neighbor in the **for** cycle is $N'(\sigma_i)$. For $j = i$, we observe that if there is a path from $(\sigma_i, \sigma_k, \dots, \sigma_j)$, then $\sigma_j \in N'(\sigma_k)$ by induction hypothesis, and $\sigma_j \in N'(\sigma_i)$ because of line 7 of the algorithm.

This means that the output graph is transitive. Moreover, the step in line 7 of the algorithm does not modify reachability, so the output graph is not bigger than the transitive closure. \square

Proof of Theorem 2.1. We first observe that the steps in lines 2 and 5 can be performed in linear time, so they have no effect on the running time of the algorithm. The “hardest” step is line 7.

In order to estimate the time needed to perform line 7, we define two different sets and we compute separately the running time for each of these sets:

$$X_\alpha := \{i : |N'(\sigma_i)| \leq n^\alpha\}$$

$$Y_\alpha := X_\alpha^C = \{i : |N'(\sigma_i)| > n^\alpha\}.$$

The running time needed to perform the instruction in line 7 for an edge (v, w) is $|N(w)|$, by using Fibonacci heaps [24]. The time needed to perform this operation for all edges (v, w) with $w \in X_\alpha$ is:

$$\sum_{(v,w) \in E \wedge w \in X_\alpha} |N(w)| \leq mn^\alpha.$$

In order to estimate the time needed to perform the check for edges (v, w) with $w \in Y_\alpha$, we observe that if $w \in Y_\alpha$, then $v \in Y_\alpha$ because $N'(v) \supseteq N'(w)$ and that $n^\alpha |Y_\alpha| \leq m$. Then, the time needed to check all these edges is at most:

$$\sum_{(v,w) \in E \wedge w \in Y_\alpha} |N(w)| \leq n|Y_\alpha|^2 \leq \frac{m^2}{n^{2\alpha-1}}.$$

Then, for each α , the total running time of the algorithm is at most $\mathcal{O}\left(\frac{m^2}{n^{2\alpha-1}} + mn^\alpha\right)$. \square

3 Hard Problems

In the previous section, we have provided truly subquadratic algorithms for two important problems. The goal of this section is the converse: proving that it is impossible to find truly subquadratic algorithms for some problems, unless SETH is false. We will provide the context under which all these reductions fall in, and in the last part we will prove them.

The starting point of our reductions is an “artificial” variation of k -SAT which is quadratic-time solvable, but not solvable in $\mathcal{O}(n^{2-\epsilon})$ unless SETH is false.

Problem: k -SAT*.
Input: two sets of variables $\{x_i\}$, $\{y_j\}$ of the same size, a set C of clauses over these variables, such that each clause has at most size k , the set of possible evaluations of x_i and the set of possible evaluations of $\{y_j\}$.
Output: **True** if there is an evaluation of all variables that satisfies all clauses, **False** otherwise.

It should be noticed that this problem differs from the classic one only by the input size. This way, a quadratic-time algorithm exists (trying all possible evaluations). However, an algorithm running in $\mathcal{O}(n^{2-\epsilon})$ with ϵ not depending on k would imply an algorithm solving k -SAT in $\mathcal{O}(2^{\frac{n}{2}(2-\epsilon)}) = \mathcal{O}((2^{\frac{2-\epsilon}{2}})^n)$, and this is against SETH.

After defining the “starting” problem, we need to define reductions. We will use quasilinear reductions from one problem to another.

Definition 3.1. A quasilinear Karp reduction from a problem \mathcal{P} to problem \mathcal{Q} is a function Φ from instances of \mathcal{P} to instances of \mathcal{Q} verifying for every instance I of \mathcal{P} :

- $\Phi(I)$ can be computed in time $\tilde{\mathcal{O}}(s(I))$, where $s(i)$ is the size of input I ;¹
- I and $\Phi(I)$ have the same output.

In general, if the output is not boolean, we will require a linear-time computable function that transforms the output of $\Phi(I)$ into the output of I . If \mathcal{P} is reducible to \mathcal{Q} , we will say $\mathcal{P} \leq_{ql} \mathcal{Q}$.

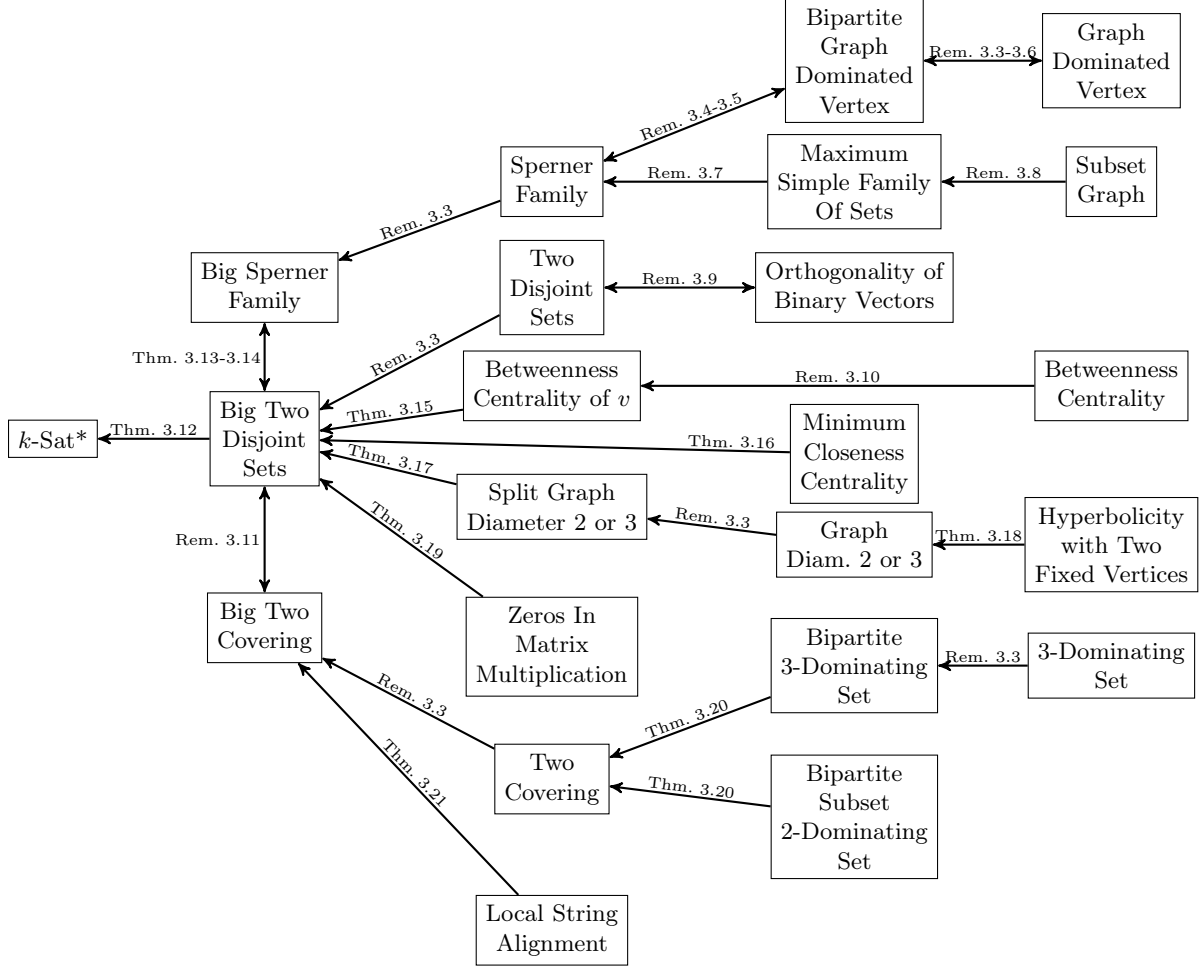


Figure 1: A scheme of the reductions provided by this article.

Remark 3.2. If $\mathcal{P} \leq_{ql} \mathcal{Q}$ and there is an algorithm solving \mathcal{Q} in time $\tilde{\mathcal{O}}(n^{2-\epsilon})$ for some ϵ , then \mathcal{P} can be solved in time $\tilde{\mathcal{O}}(n^{2-\epsilon})$.

The reductions that we will provide are summarized in Figure 1 (the definition of all problems is given in Appendix A). The previous remark implies that an $\mathcal{O}(n^{2-\epsilon})$ algorithm for any of these problems would falsify SETH. All proofs of those reductions will be provided by the next subsection.

3.1 Proof of Reductions

In this subsection, we will prove all the reductions in Figure 1. We have divided these reductions in remarks and theorem, depending on how intricate those constructions are. We will start by providing the proof of remarks, in the order in which they appear in Figure 1.

Remark 3.3. For each problem, \mathcal{P} , $\text{BIG}\mathcal{P} \leq_{ql} \mathcal{P}$, since the instances of $\text{BIG}\mathcal{P}$ are a subset of the instances of \mathcal{P} and the required output is the same (the function Φ in the definition of reductions is the identity). The same argument proves that $\text{BIPGDOMINATEDVERTEX} \leq_{ql} \text{GRAPHDOMINATEDVERTEX}$, that $\text{BIPARTITE3DOMINATINGSET} \leq_{ql} \text{3DOMINATINGSET}$ and that $\text{SPLITGRAPHDIAMETER2OR3} \leq_{ql} \text{GRAPHDIAMETER2OR3}$.

Remark 3.4. $\text{HYPERGRAPHDOMINATEDEDGE} \leq_{ql} \text{BIPGDOMINATEDVERTEX}$: each hypergraph (V, E) can be transformed to a biparted graph (V, W, A) where $W = E$ and $A(v, e)$ holds if $v \in e$

¹By $\tilde{\mathcal{O}}(f(n))$ we mean $\mathcal{O}(f(n) \log^k n)$ for some fixed k .

(hypergraph edges are seen as sets of vertices). It is clear that an edge e dominates e' in the hypergraph if and only if the corresponding element in W does. The only problem is that two vertices $v, v' \in V$ might also dominate each other: to this purpose, we add to W another copy of V named V' and we add to E all edges (v, v') from V to V' .

Remark 3.5. $\text{BIPGDOMINATEDVERTEX} \leq_{ql} \text{HYPERGRAPHDOMINATEDEDGE}$: starting from the bipartite graph (V, V', A) , create two hypergraphs (V, E) and (V', E') . The edges are defined as follows: for each $v \in V$ we add $N(v)$ to E' (since $N(v)$ is contained in V'), and likewise for each $v' \in V'$ we add $N(v')$ to E . This way, if an edge $v \in V$ dominates an edge $w \in V$, the corresponding edges in V' in the second hypergraph dominate each other. Conversely, if the dominated vertex is in V' , there is a dominated edge in the first hypergraph. In order to conclude the reduction, it is enough to consider the disjoint union of (V, E) and (V', E') .

Remark 3.6. $\text{GRAPHDOMINATEDVERTEX} \leq_{ql} \text{BIPGDOMINATEDVERTEX}$: given a graph $G = (V, E)$, let us construct a bipartite graph $G' = (V, V', A)$ where V and V' are copies of V and $A(v, v')$ holds if $v \in V$, $v' \in V'$ and $E(v, v')$ holds in G . It is clear that this construction preserves domination.

Remark 3.7. $\text{SPERNERFAMILY} \leq_{ql} \text{MAXIMALELEMENTSFAMILY}$, since a family \mathcal{C} of sets has a dominated set if and only if it is not simple, if and only if the maximum simple family of sets is \mathcal{C} .

Remark 3.8. $\text{MAXIMALELEMENTSFAMILY} \leq_{ql} \text{SUBSETGRAPH}$, since the maximum simple family of sets is the set of sources of the subset graph.

Remark 3.9. $\text{ORTHOGONALITYBINARYVECTORS} =_{ql} \text{TWODISJOINTSETS}$: it is enough to choose an ordering on X and to code a set $C \subseteq X$ as a vector of length $|X|$ having 1 at place x if $x \in C$, 0 otherwise. Two such vectors are orthogonal if and only if the set of ones in these two vectors are disjoint (we are assuming that the ring over which the vector are considered has characteristic 0, that is, in that ring, $1 + 1 + \dots + 1$ is never 0). We are also assuming a “clever” data structure to store binary vectors, that is, no space is needed to memorize zeros.

Remark 3.10. $\text{BETWEENNESSCENTRALITYVERTEX} \leq_{ql} \text{BETWEENNESSCENTRALITY}$, because computing the betweenness centrality of all vertices is of course harder than computing the betweenness centrality of a single vertex.

Remark 3.11. $\text{BIGTWOISJOINTSETS} =_{ql} \text{BIGTWOCOVERING}$, because $C, C' \in \mathcal{C}$ are disjoint if and only if C^C, C'^C cover X : it is enough to define $\Phi(X, \mathcal{C}) = (X, \bar{\mathcal{C}})$, where $\bar{\mathcal{C}} = \{C^C : C \in \mathcal{C}\}$. Note that the input size in the reduction can increase only by a logarithmic factor, since the size of X is $\mathcal{O}(\log^k(|\mathcal{C}|))$.

We may now turn to the proof of the theorems. Again, the theorems are sorted according to Figure 1.

Theorem 3.12. *For each k , $k\text{-SAT}^* \leq_{ql} \text{BIGTWOISJOINTSETS}$.*

Proof. Let X_1 be the set of the possible evaluations of $\{x_i\}$, X_2 the set of possible evaluations of $\{y_j\}$, C the set of clauses of an instance I of $k\text{-SAT}^*$. We define $\Phi(I) = (X, \mathcal{C})$, where $X = C \cup \{x_1, x_2\}$, and \mathcal{C} is the collection made by sets of clauses not satisfied by an assignment in X_1 or X_2 (and x_1, x_2 are used to distinguish between assignments in X_1 and X_2).

More formally, $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$, $\mathcal{C}_1 := \{\{x_1\} \cup \{c \in C : x \not\models c\} : x \in X_1\}$ and similarly $\mathcal{C}_2 := \{\{x_2\} \cup \{c \in C : x \not\models c\} : x \in X_2\}$. This way, the size of $\Phi(I)$ is linear in the size of I . Moreover, it is possible to compute Φ by analyzing all sets of evaluation of variables one by one, and for each of them check which clauses are verified. For each evaluation in X_1 or X_2 , the checking time is proportional to the number of clauses, which is at most $\mathcal{O}(\log^k(n))$, where n is the input size of $k\text{-SAT}^*$.

It remains to prove that the output of the problem is preserved: we will prove that there is a bijection between the pairs of disjoint sets and the satisfying assignments of the formula of $k\text{-SAT}^*$.

In particular, two sets that are both in \mathcal{C}_1 or both in \mathcal{C}_2 cannot be disjoint because of x_1 and x_2 . As a consequence, two disjoint sets correspond to an evaluation of all variables: the evaluation satisfies ϕ if and only if for each clause there is a variable contained in the evaluation if and only if there is no clause contained in both sets. \square

Theorem 3.13. $\text{BIGTWO} \text{DISJOINTSETS} \leq_{ql} \text{BIGSPERNERFAMILY}$.

Proof. Consider an instance $I = (X, \mathcal{C})$ of $\text{BIGTWO} \text{DISJOINTSETS}$. First of all, we define $\Phi'(I) = (X, \mathcal{C}')$, where $\mathcal{C}' = \mathcal{C} \cup \bar{\mathcal{C}}$, and $\bar{\mathcal{C}} := \{C^C : C \in \mathcal{C}\}$ (which is not the correct definition, but we will see how to adapt it).

If we find two sets $C \in \mathcal{C}, C' \in \bar{\mathcal{C}}$ such that $C \subseteq C'$, we know that C and C'^C are disjoint and in \mathcal{C} , so we have found a solution. However, we might also find two sets $C \subseteq C', C' \in \mathcal{C}, C \subseteq C'$ with $C \in \bar{\mathcal{C}}$ and $C' \in \mathcal{C}, C \subseteq C', C' \in \bar{\mathcal{C}}$. The remaining part of the proof slightly modifies the set X and the collection \mathcal{C}' in order to avoid such cases.

The first problem can be solved by defining $k := \lceil \log_2(|\mathcal{C}|) \rceil$, and adding two sets $Y = \{y_1, \dots, y_k\}$ and $Z = \{z_1, \dots, z_k\}$ to X . In particular, we add Y and Z to each set in $\bar{\mathcal{C}}$ and we add to each element $C \in \mathcal{C}$ some y_i and some z_j , so that no element of \mathcal{C} can dominate another element in \mathcal{C} (for example, we may associate each set C with a unique binary number with k bits, and code this number using y_i as zeros and z_j as ones). In order to solve the second problem, it is enough to make the same construction adding new sets Y' and Z' of logarithmic size, and use them to uniquely code any element in $\bar{\mathcal{C}}$. None of the elements in Y' and Z' is added to elements in \mathcal{C} , and this also solves the third problem. \square

Theorem 3.14. $\text{BIGSPERNERFAMILY} \leq_{ql} \text{BIGTWO} \text{DISJOINTSETS}$.

Proof. Let us consider an instance $I = (X, \mathcal{C})$ of BIGDOMINATEDSET and let us define $\Phi(I) = (X \cup x_1, x_2, \mathcal{C}')$, where $\mathcal{C}' := \mathcal{C}_1 \cup \mathcal{C}_2$, $\mathcal{C}_1 := \{C \cup x_1 : C \in \mathcal{C}\}$ and $\mathcal{C}_2 := \{C \cup x_2 : C \in \mathcal{C}\}$.

If C_1 and C_2 are disjoint sets in \mathcal{C}' , one of them must be in \mathcal{C}_1 and the other in \mathcal{C}_2 (because of x_1 and x_2). As a consequence, there are two disjoint sets $C_1 \in \mathcal{C}_1, C_2 \in \mathcal{C}_2$ in $\Phi(I)$ if and only if $C_1 \cap X \in \mathcal{C}, C_2 \cap X \in \mathcal{C}$ are disjoint if and only if $C_1 \cap X \subseteq C_2 \cap X$ (and this means that \mathcal{C} contains a set dominating another). \square

Theorem 3.15. $\text{BIGTWO} \text{DISJOINTSETS} \leq_{ql} \text{BETWEENNESSCENTRALITYVERTEX}$.

Proof. Let us consider an instance $I = (X, \mathcal{C})$ of $\text{BIGTWO} \text{DISJOINTSETS}$, and let us construct a graph $G = (V, E)$ as follows:

- $V := \{y\} \cup \{x\} \cup \mathcal{C}_x \cup X \cup \mathcal{C}_y$, where X is the ground set of I and $\mathcal{C}_x, \mathcal{C}_y$ are two identical copies of \mathcal{C} (in the graph, $y, x, \mathcal{C}_x, X, \mathcal{C}_y$ will somehow resemble a cycle).
- all pairs of vertices in V' are connected;
- vertex x is connected to y and to each vertex in \mathcal{C}_x ;
- vertex y is connected to x and to each vertex in \mathcal{C}_y ;
- connections between \mathcal{C}_x and X and connections between \mathcal{C}_y and X are made according to the \in -relation.

The input vertex of our problem is x .

We observe that the graph is a big “cycle” made by five “parts”: $y, x, \mathcal{C}_x, X, \mathcal{C}_y$. Moreover, it is possible to move from one part to any vertex in the next part in one step (except if we start or arrive in X , and in that case we need at most two steps). As a consequence, no shortest path can be longer than 3.

This proves that any shortest path in the sum passing through x must be of one of the following forms:

- a path from y to \mathcal{C}_x ;
- a path from \mathcal{C}_x to \mathcal{C}_y ;
- a path from y to X

We note that the third case never occurs, because there always exists a path from y to any vertex in X of length 2. The first case occurs for each vertex in V_x , and no other shortest path exists from y to vertices in \mathcal{C}_x : these vertices contribute to the sum by $|\mathcal{C}|$. Finally, the second case occurs only if and only if there is a pair of vertices in \mathcal{C}_y and \mathcal{C}_x having no path of length 2, that is, two disjoint sets in \mathcal{C} .

This proves that the betweenness of x is bigger than $|\mathcal{C}|$ if and only if there are two disjoint sets in \mathcal{C} . \square

Theorem 3.16. $\text{BIGTWO} \text{DISJOINTSETS} \leq_{qt} \text{MINIMUMCLOSENESSCENTRALITY}$.

Proof. Instead of minimizing the closeness centrality, we will try to maximize the *farness*, which is the inverse of the closeness centrality, that is, the sum of all distances from v to another vertex. In our construction, we will build a graph where the vertices with biggest farness correspond to sets in \mathcal{C} , and the value of the farness does not depend on the corresponding set, if this set is not disjoint to any other set. If this latter condition is not satisfied, then the farness of the vertex is bigger. In particular, let us consider an instance $I = (X, \mathcal{C})$ of $\text{BIGTWO} \text{DISJOINTSETS}$ let us assume $X \not\subseteq \mathcal{C}$, and let us build a graph in the following way:

- $V = V_1 \cup V'_1 \cup V_2 \cup V_3$, where V_1 and V'_1 are two disjoint copies of X , $V_2 = \mathcal{C}$ and $V_3 = \{(x, C) \in X \times \mathcal{C} : x \notin C\}$;
- $V_1 \cup V'_1$ is a clique;
- for $x \in V_1 \cup V'_1$ and $C \in V_2$, there is an edge from x to C if and only if $x \in C$;
- for each $(x, C) \in V_3$ and $C' \in V_2$, there is a link between these vertices if and only if $C = C'$.

Claim: the vertex with maximum farness is in V_3 .

Proof of claim. For each vertex $v \in V_2$, consider a vertex $w \in V_3$ linked to v . It is clear that all shortest paths from w to any other vertex must pass through v (which is the only vertex linked to w). This means that the farness of w is bigger than the farness of v .

For each vertex $v \in V_1 \cup V'_1$, let us consider a vertex $w \in V_2$ linked to v . The only vertices which are closer to w than to v are the vertices in V_3 attached to w , because each other neighbor of w is a neighbor of v . These vertices influence the farness of w by $|X| - |C|$, where C is the set corresponding to w . However, there are $2(|X| - |C|)$ vertices in $V_1 \cup V'_1$ linked to v and not to w (the elements not in C): this proves that the farness of w is bigger than the farness of v . \square

At this point, let us consider the farness of vertices in V_3 . In particular, let (x, C) be an element of V_3 such that $C \cap C' \neq \emptyset$ for each C' : the farness of (x, C) can be exactly computed by considering the classes of vertices in Table 1.

Before computing the farness of (x, C) , we compute $\sum_{C' \neq C} |X| - |C'| = (|\mathcal{C}| - 1)|X| - \sum_{C' \neq C} |C'| = (|\mathcal{C}| - 1)|X| - \sum_{C' \in \mathcal{C}} |C'| + |C|$. The farness of (x, C) is then:

$$\begin{aligned}
& 4|C| + 6(|X| - |C|) + 1 + 3(|\mathcal{C}| - 1) + 2(|X| - |C|) + 4 \left((|\mathcal{C}| - 1)|X| - \sum_{C' \in \mathcal{C}} |C'| + |C| \right) = \\
& = 4|C| + 8|X| - 8|C| + 1 + 3|\mathcal{C}| - 3 + 4(|\mathcal{C}| - 1)|X| - 4 \sum_{C' \in \mathcal{C}} |C'| + 4|C| =
\end{aligned}$$

Table 1: The distance from (x, C) to another vertex

Set	Kind of vertex	Number	distance from (x, C)
$V_1 \cup V'_1$	vertex in C	$2 C $	2
$V_1 \cup V'_1$	vertex outside C	$2(X - C)$	3
V_2	C	1	1
V_2	$C' \neq C$	$ C - 1$	3
V_3	(x', C)	$ X - C $	2
V_3	$(x', C'), C' \neq C$	$\sum_{C' \neq C} X - C' $	4

$$= 4|C||X| - 4 \left(\sum_{C' \in \mathcal{C}} |C'| \right) + 3|C| + 4|X| - 2.$$

Note that this value does not depend on the particular (x, C) chosen (this was the main goal of our construction). It is clear that if $C \cap C' = \emptyset$, then the farness of each vertex (x, C) and (x, C') is bigger than the value previously computed.

As a consequence, there are two disjoint sets if and only if in the whole graph there is a vertex with farness bigger than $4|C||X| - 4(\sum_{C' \in \mathcal{C}} |C'|) + 3|C| + 4|X| - 2$, and both this value and the underlying graph can be computed in linear time. \square

Theorem 3.17. $\text{BIGTWO} \text{DISJOINTSETS} \leq_{ql} \text{SPLITGRAPHDIAMETER2OR3}$.

Proof. Given an input $I = (X, \mathcal{C})$ of $\text{BIGTWO} \text{DISJOINTSET}$, construct a split graph $G = (X \cup \mathcal{C}, E)$, where each pair in X is connected, and for each set $C \in \mathcal{C}$ we add an edge from C to its elements.

Since each vertex is at distance 1 from X , the diameter is 2 or 3: it is 3 if and only if there exist two different vertices $C, C' \in \mathcal{C}$ with no common neighbor. It is clear that this happens if and only if C, C' are disjoint. \square

Theorem 3.18. $\text{GRAPHDIAMETER2OR3} \leq_{ql} \text{HYPERBOLICITYWITH2FIXEDVERTICES}$.

Proof. Let $G = (V, E)$ be an input graph for the GRAPHDIAMETER2OR3 problem. The corresponding graph for the $\text{HYPERBOLICITYWITH2FIXEDVERTICES}$ problem is $H = (V', E')$, where $V' = \{x\} \cup V_x \cup \tilde{V} \cup V_y \cup \{y\}$, where V_x , \tilde{V} and V_y are disjoint copies of V . Edges in E' are defined as follows:

- x is connected to every vertex in V_x and y is connected to every vertex in V_y ;
- corresponding vertices in V_x and V and corresponding vertices in V and V_y are connected;
- if (v, w) is an edge of G , then the copies of v and w in \tilde{V} are linked.

In the instance of the $\text{HYPERBOLICITYWITH2FIXEDVERTICES}$ problem we ask if the maximum hyperbolicity of a quadruple containing x and y is bigger than 2. We will prove that this holds if and only if the diameter of G is bigger than 2.

Let us first consider quadruples with vertices x, y, v, w with $v, w \in \tilde{V}$. In these quadruples, $S_1 = d(x, y) + d(v, w) = 4 + d(v, w)$, $S_2 = S_3 = 4$, since the distance from x and y to any vertex in \tilde{V} is 2. The hyperbolicity of such a quadruple is then equal to $d(v, w)$, which reaches the maximum if v, w are a diametral pair. We conclude that if we restrict to such quadruples, then the maximum hyperbolicity equals the diameter.

It remains only to prove that all other quadruples have smaller hyperbolicity. If $v, w \in V_x$ (or $v, w \in V_y$), then $d(v, w) = 2$ by passing through x , and $S_1 = d(x, y) + d(v, w) = 6$, $S_2 =$

$d(x, v) + d(y, w) = 4$ and similarly $S_3 = 4$. The hyperbolicity is then 2. Otherwise, if $v \in V_x$ and $w \notin V_x$, $S_2 = d(x, w) + d(y, v) = 3 + d(y, v) \geq 5$ and $S_3 = d(x, v) + d(y, w) = 1 + d(y, w) = 1 + 4 - d(y, v) = 5 - d(y, v) \leq 3$. As a consequence, if vertex v is moved to the corresponding vertex in \tilde{V} , S_3 can only increase by one and S_1 can only decrease by one. Moreover, S_2 decreases by one, meaning that, if S'_i are the new values, $S'_1 - S'_2 \geq S_1 - 1 - (S_2 - 1) = S_1 - S_2$, and the hyperbolicity of the 4-tuple is increased. The proof for all other cases are symmetrical. \square

Theorem 3.19. $\text{BIGTWO} \text{DISJOINTSETS} \leq_{ql} \text{ZEROMATRIXMULTIPLICATION}$.

Proof. Let (X, \mathcal{C}) be an instance of the $\text{BIGTWO} \text{DISJOINTSETS}$ problem. Consider the $|X| \times |\mathcal{C}|$ matrix M containing 1 in place (x, C) if and only if $x \in C$. It is clear that $M^T M$ contains a zero in place C, C' if and only if $C \cap C' = \emptyset$. If we want the matrix to be square, it is enough to add $|\mathcal{C}| - |X|$ empty lines after the first $|X|$ lines (this does not change the input size as long as the matrix is stored as an adjacency list). \square

Theorem 3.20. $\text{TWOCOVERING} \leq_{ql} \text{BIPARTITE3DOMINATINGSET}$.

$\text{TWOCOVERING} \leq_{ql} \text{BIPARTITESUBSET2DOMINATINGSET}$.

Proof. Let $I = (X, \mathcal{C})$ be an instance of TWOCOVERING , and let us create a bipartite graph $G = (V, W, E)$ with $V = X$, $W = \mathcal{C}$ and E defined as the set of pairs (x, C) such that $x \in C$. Then, each 2-covering of X with sets in \mathcal{C} corresponds to a pair of vertices of W that covers V .

This way, we proved that $\text{TWOCOVERING} \leq_{ql} \text{BIPARTITESUBSET2DOMINATINGSET}$, by choosing V as the subset to cover.

In order to prove that $\text{TWOCOVERING} \leq_{ql} \text{BIPARTITE3DOMINATINGSET}$, we add another vertex $v_0 \in V$ connected to any $w \in W$: clearly, if there is a 2-covering of X , there is a dominating triple in G , which is the 2-covering and v .

Viceversa, if there is a dominating triple, one of the vertices of the triple must be in V : the other two vertices form a dominating pair (if only one of them is in W , the corresponding set is the whole X). \square

Theorem 3.21 (based on Lemma 1 in [2]). $\text{BIGTWO} \text{COVERING} \leq_{ql} \text{LOCALSTRINGALIGN}$.

Proof. Let us consider an instance (X, \mathcal{C}) of problem $\text{BIGTWO} \text{COVERING}$, assuming $X = 0, \dots, k-1$. We code each element $C \in \mathcal{C}$ into a binary string, which is a concatenation of k chunks of the form $01?01$. The $?$ in the i -th chunk is 1 if $i \in C$, 0 otherwise. We then concatenate all strings for each element $C \in \mathcal{C}$, separated with the string 000 . This way, we constructed the first string.

The binary strings forming the second string are similarly made, but there is a difference in chunks, which are of the form $*1?*1$, where $*$ is the character that may be paired with any other character, and $?$ is $*$ if $i \in C$, 1 otherwise. these strings are concatenated with 111 as separator.

We want to prove that two sets C, C' cover \mathcal{C} if and only if the longest common substring of the two strings has length $5k$.

First, let us suppose that there is a common string of length $5k$: we claim first that no separator can occur in this string. This happens because in the first string there is never the sequence 111 and in the second string there is no 000 . This means that the two equivalent strings of length $5k$ must correspond to two subsets, eventually translated by $-2, -1, 1$ or 2 positions. The sum of the two translation is then smaller than 4 in absolute value.

However, with a case by case analysis, it is possible to exclude all such translations. For instance, a -1 translation would imply that the 6th element in the first string (0) matches the 5th element in the second string (1), and a $+2$ translation would imply that element $5i + 2$ (which is 0 if $i \notin C$) would match element $5i + 4$ in the second string, which is 1.

We concluded that a common substring of length $5k$ must be made by two pieces corresponding to a set. Now it is easy to see that places $5i, 5i + 1, 5i + 3, 5i + 4$ always correspond,

while places $5i + 2$ correspond if and only if element i is in at least one of the two sets. This concludes the proof. □

4 Conclusions and Open Problems

In this paper, we have analyzed many results on quadratic-time problems. We have proved that two of these problems are solvable in truly subquadratic time (recognizing transitive graphs and comparability graphs), and we have provided hardness results for many others. This work can be seen as a starting point to develop many more reductions and include inside this class many new problems.

For instance, it would be really interesting to link these results with all existing reductions on the 3SUM problem: until now, we have not been able to link it with SETH. However, it is possible to link it with other problems: for example, it is known that the local alignment of strings problem is harder than 3SUM [2].

Another open problem deals with the radius of graphs: this measure is similar to the diameter, but it looks somehow “easier” to compute, for example because the radius of chordal graphs is linear-time computable [13]. The question is whether also this problem can be inserted in our class of quadratic-time hard problems, or a truly subquadratic algorithm exists.

Among other problems that are not in our class, but have no truly subquadratic time algorithm, we find the computation of the transitive reduction of a directed graph (a “converse” of the transitive closure), finding maximum flows in networks or finding maximum matchings in bipartite weighted graphs. All these problems are defined in Appendix A.

References

- [1] Amir Abboud and Virginia V. Williams. Popular conjectures imply strong lower bounds for dynamic problems. *FOCS*, 2014.
- [2] Amir Abboud, Virginia V. Williams, and Oren Weimann. Consequences of Faster Alignment of Sequences. *ICALP*, 2014.
- [3] David A. Bader, Shiva Kintali, Kamesh Madduri, and Milena Mihail. Approximating betweenness centrality. *The 5th Workshop on Algorithms and Models for the Web-Graph*, 2007.
- [4] Ilya Baran, Erik D. Demaine, and Mihai Patrascu. Subquadratic algorithms for 3SUM. *Algorithmica*, 50(4):584–596, September 2008.
- [5] Alex Bavelas. Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America*, 22:725–730, 1950.
- [6] Christian Bey, Konrad Engel, Gyula O. H. Katona, and Uwe Leck. On the average size of sets in intersecting sperner families. *Discrete Math.*, 257(2-3):259–266, November 2002.
- [7] Guy E. Blelloch, Virginia Vassilevska, and Ryan Williams. A new combinatorial approach for sparse graph problems. *Automata, Languages and Programming*, 2008.
- [8] Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter Kusters, Andrea Marino, and Frank Takes. On the Solvability of the Six Degrees of Kevin Bacon Game - A Faster Graph Diameter and Radius Computation Method. In *Fun with Algorithms*, pages 57–68, 2014.
- [9] Ulrik Brandes. A faster algorithm for betweenness centrality*. *The Journal of Mathematical Sociology*, 25(2):163–177, June 2001.
- [10] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*. Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.
- [11] Yangjun Chen. A new algorithm for transitive closures and computation of recursion in relational databases. *Information Visualization, 2003. IV 2003. Proceedings. Seventh International Conference on Information Visualization*, 01(242523), 2003.
- [12] Victor Chepoi, Fedor F. Dragan, Bertrand Estellon, Michel Habib, and Yann Vaxès. Diameters, centers, and approximating trees of delta-hyperbolicgeodesic spaces and graphs. *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 59–68, 2008.
- [13] Victor Chepoi and Feodor F. Dragan. A linear-time algorithm for finding a central vertex of a chordal graph. In *ESA*, pages 159–170, 1994.
- [14] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. Computing Classic Closeness Centrality, at Scale. Technical report, 2014.
- [15] Nathann Cohen, David Coudert, and Aurélien Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Technical Report September, 2012.
- [16] Pilu Crescenzi, Roberto Grossi, Michel Habib, Leonardo LANZI, and Andrea Marino. On computing the diameter of real-world undirected graphs. *Theoretical Computer Science*, 514:84–95, November 2013.

- [17] Ben Dushnik and E. W. Miller. Partially Ordered Sets. *American Journal of Mathematics*, 63(3):600–610, 1941.
- [18] Nick Edmonds, Torsten Hoefler, and Andrew Lumsdaine. A space-efficient parallel algorithm for computing betweenness centrality in distributed memory. *2010 International Conference on High Performance Computing*, pages 1–10, December 2010.
- [19] Friedrich Eisenbrand and Fabrizio Grandoni. On the complexity of fixed parameter clique and dominating set. *Theoretical Computer Science*, 326(1-3):57–67, October 2004.
- [20] Amr Elmasry. The Subset Partial Order: Computing and Combinatorics. *ANALCO*, pages 27–33, 2010.
- [21] Konrad Engel. *Sperner Theory*. Cambridge Solid State Science Series. Cambridge University Press, 1997.
- [22] Jeff Erickson. New lower bounds for Hopcroft’s problem. *Discrete & Computational Geometry*, (November):1–11, 1996.
- [23] Hervé Fournier, Anas Ismail, and Antoine Vigneron. Computing the Gromov hyperbolicity of a discrete metric space. *arXiv preprint arXiv:1210.3323*, pages 1–6, 2012.
- [24] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.
- [25] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 1977.
- [26] Alla Goralcíkova and Vaclav Koubek. A reduct-and-closure algorithm for graphs. *Mathematical Foundations of Computer Science 1979*, pages 301–307, 1979.
- [27] Mikhael Gromov. *Hyperbolic groups*. Springer, 1987.
- [28] Michel Habib, R McConnell, Christophe Paul, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234:59–84, 2000.
- [29] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which Problems Have Strongly Exponential Complexity? *Journal of Computer and System Sciences*, 63(4):512–530, December 2001.
- [30] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.
- [31] James King. A survey of 3SUM-hard problems. Technical report, 2004.
- [32] Dexter Kozen. *Design and analysis of algorithms*. Texts and monographs in computer science. Springer, 1992.
- [33] Vito Latora and Massimo Marchiori. A measure of centrality based on network efficiency. *New Journal of Physics*, (February 2008), 2007.
- [34] Jakub Lkacki. Improved Deterministic Algorithms for Decremental Transitive Closure and Strongly Connected Components. In *Proceedings of the Twenty-second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA ’11, pages 1438–1445. SIAM, 2011.
- [35] Ross M. McConnell and Jeremy Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999.

- [36] Mark Newman. *Networks: An Introduction*. Oxford University Press, 2010.
- [37] Mihai Patrascu and Ryan Williams. On the possibility of faster SAT algorithms. *ACM-SIAM Symposium on Discrete Algorithms*, 2010.
- [38] Paul Pritchard. On computing the subset graph of a collection of sets. *Journal of Algorithms*, (October):1–14, 1999.
- [39] Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing - STOC '13*, page 515, 2013.
- [40] Liam Roditty and Virginia Vassilevska Williams. Approximating the diameter of a graph. *arXiv*, (822), 2014.
- [41] Frank Takes and Walter Kusters. Computing the Eccentricity Distribution of Large Graphs. *Algorithms*, 6(1):100–118, February 2013.
- [42] Frank W. Takes and Walter A. Kusters. Determining the diameter of small world networks. *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11*, pages 1191–1196, 2011.
- [43] Ryan Williams and Huacheng Yu. Finding orthogonal vectors in discrete structures. *SODA*, 2014.
- [44] Virginia Vassilevska Williams. Multiplying Matrices Faster Than Coppersmith-winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA, 2012. ACM.
- [45] Virginia Vassilevska Williams and Ryan Williams. Subcubic Equivalences between Path, Matrix and Triangle Problems. *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654, October 2010.
- [46] Yaokun Wu and Chengpeng Zhang. Hyperbolicity and chordality of a graph. *the electronic journal of combinatorics*, 18:1–22, 2011.
- [47] Daniel M. Yellin and Charanjit S. Jutla. Finding Extremal Sets in Less Than Quadratic Time. *Inf. Process. Lett.*, 48(1):29–34, 1993.

Appendix A Problem Definitions

In this section, we will precisely define all the problems we are dealing with.

Hard Problems

Problem: BETWEENNESSCENTRALITY.

Input: a graph $G = (V, E)$.

Output: the betweenness centrality of each vertex v of G , that is,

$$\sum_{v \neq s \neq t \in V} \frac{\text{number of shortest paths from } s \text{ to } t \text{ through } v}{\text{number of shortest paths from } s \text{ to } t}.$$

Problem: BETWEENNESSCENTRALITYVERTEX.

Input: a graph $G = (V, E)$ and a vertex $v \in V$.

Output: the betweenness centrality of v , that is,

$$\sum_{v \neq s \neq t \in V} \frac{\text{number of shortest paths from } s \text{ to } t \text{ through } v}{\text{number of shortest paths from } s \text{ to } t}.$$

Problem: BIGSPERNERFAMILY.

Input: a set X and a collection \mathcal{C} of subsets of X such that $|X| < \log^k(|\mathcal{C}|)$ for some k .

Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $C \subseteq C'$, **False** otherwise.

Problem: BIGTWOCOVERING.

Input: a set X and a collection \mathcal{C} of subsets of X such that $|X| < \log^k(|\mathcal{C}|)$ for some k .

Output: **True** if there are two sets $C, C' \in \mathcal{C}$ such that $X = C \cup C'$, **False** otherwise.

Problem: BIGTWO disjoint SETS.

Input: a set X and a collection \mathcal{C} of subsets of X such that $|X| < \log^k(|\mathcal{C}|)$ for some k .

Output: **True** if there are two disjoint sets $C, C' \in \mathcal{C}$, **False** otherwise.

Problem: BIPARTITE3DOMINATINGSET.

Input: a bipartite graph $G = (V, E)$.

Output: a triple v, w, x such that $V = N(v) \cup N(w) \cup N(x) \cup \{v, w, x\}$, if it exists.

Problem: BIPARTITESUBSET2DOMINATINGSET.

Input: a graph $G = (V, E)$ and a subset $V' \subseteq V$.

Output: a pair v, w such that $V' = N(v) \cup N(w) \cup \{v, w\}$, if it exists.

Problem: BIPGDOMINATEDVERTEX.

Input: a bipartite graph (V_1, V_2, E) .

Output: **True** if there are vertices v, w such that $N(v) \supseteq N(w)$, **False** otherwise.

Problem: 3DOMINATINGSET.

Input: a graph $G = (V, E)$.

Output: a triple v, w, x such that $V = N(v) \cup N(w) \cup N(x) \cup \{v, w, x\}$, if it exists.

Problem: GRAPHDIAMETER2OR3.

Input: a graph G .

Output: **True** if G has diameter 2, **False** otherwise.

Problem: GRAPHDOMINATEDVERTEX.

<i>Input:</i>	a graph (V, E) .
<i>Output:</i>	True if there are vertices v, w such that $N(v) \supseteq N(w)$, False otherwise.
<i>Problem:</i>	HYPERBOLICITYWITH2FIXEDVERTICES.
<i>Input:</i>	a graph $G = (V, E)$ and two vertices x, y .
<i>Output:</i>	the maximum hyperbolicity of a quadruple x, y, v, w , where the hyperbolicity of a quadruple is $S_1 - S_2$, where S_1 is the maximum sum among $d(x, y) + d(v, w), d(x, v) + d(y, w), d(x, w) + d(y, v)$, and S_2 the second maximum sum.
<i>Problem:</i>	k -SAT*.
<i>Input:</i>	two sets of variables $\{x_i\}, \{y_j\}$ of the same size, a set C of clauses over these variables, such that each clause has at most size k , the set of possible evaluations of x_i and the set of possible evaluations of $\{y_j\}$.
<i>Output:</i>	True if there is an evaluation of all variables that satisfies all clauses, False otherwise.
<i>Problem:</i>	LOCALSTRINGALIGN.
<i>Input:</i>	two binary strings with a symbol $*$ that may replace any character.
<i>Output:</i>	The longest common substring of the two strings.
<i>Problem:</i>	MAXIMALELEMENTSFAMILY.
<i>Input:</i>	a set X and a collection \mathcal{C} of subsets of X .
<i>Output:</i>	the maximum simple family of subsets of X (simple means without inclusion).
<i>Problem:</i>	MINIMUMCLOSENESSCENTRALITY.
<i>Input:</i>	a graph $G = (V, E)$ and a threshold σ .
<i>Output:</i>	True if there exists a vertex with closeness centrality smaller than σ , False otherwise. The closeness centrality of a vertex is defined as $\frac{1}{\sum_{w \in V} d(v, w)}$, where $d(v, w)$ is the distance between vertices v and w .
<i>Problem:</i>	ORTHOGONALITYBINARYVECTORS.
<i>Input:</i>	a collection of binary vectors.
<i>Output:</i>	True , if there are two orthogonal vectors, False otherwise.
<i>Problem:</i>	SPERNERFAMILY.
<i>Input:</i>	a set X and a collection \mathcal{C} of subsets of X .
<i>Output:</i>	True if there are two sets $C, C' \in \mathcal{C}$ such that $C \subseteq C'$, False otherwise.
<i>Problem:</i>	SPLITGRAPHDIAMETER2OR3.
<i>Input:</i>	a split graph G .
<i>Output:</i>	True if G has diameter 2, False otherwise.
<i>Problem:</i>	SUBSETGRAPH.
<i>Input:</i>	a set X and a collection \mathcal{C} of subsets of X .
<i>Output:</i>	a tree ordering on the sets in \mathcal{C} by inclusion.
<i>Problem:</i>	TWOCOVERING.
<i>Input:</i>	a set X and a collection \mathcal{C} of subsets of X .
<i>Output:</i>	True if there are two sets $C, C' \in \mathcal{C}$ such that $X = C \cup C'$, False otherwise.
<i>Problem:</i>	TWODISJOINTSETS.

Input: a set X and a collection \mathcal{C} of subsets of X .
Output: **True** if there are two disjoint sets $C, C' \in \mathcal{C}$, **False** otherwise.

Problem: ZEROMATRIXMULTIPLICATION.
Input: two $(0 - 1)$ -matrices M, M' implemented as adjacency lists.
Output: **True** if MM' contains a 0, **False** otherwise.

Easy Problems

Problem: COMPARABILITYRECOGNITION.
Input: an undirected graph G .
Output: **True** if there is a transitive orientation of G , **False** otherwise.

Problem: TRANSITIVECLOSURE.
Input: a directed acyclic graph G .
Output: the transitive closure of G , that is, the minimum subgraph of G such that if there is a path from a vertex v to a vertex w , $E(v, w)$ holds.

A.1 Problems not Classified Yet

Problem: NETWORKFLOW.
Input: a graph $G = (V, E)$ and two vertices v, w .
Output: the maximum flow allowed by the network from v to w .

Problem: RADIUS.
Input: a graph $G = (V, E)$.
Output: the radius of G , that is, $\min_{v \in V} \max_{w \in V} d(v, w)$.

Problem: TRANSITIVEREDUCTION.
Input: a directed acyclic graph G .
Output: the transitive reduction of G , that is, the minimum subgraph of G having the same transitive closure.

Problem: WEIGHTEDBIMAXIMUMMATCHING.
Input: a bipartite weighted graph G .
Output: a maximum matching of G .